Preprint
UCRL-JC-146794

Developing Intelligent Transportation Systems in an Integrated Systems Analysis Environment

S.M. Aceves and E.E. Paddack

This article was submitted to
9th World Congress on Intelligent Transportation Systems, Chicago, Illinois, October 14-18, 2002

U.S. Department of Energy
Lawrence Livermore National Laboratory

January 15, 2002

Developing Intelligent Transportation Systems in an Integrated Systems Analysis Environment

Salvador M. Aceves and Erma Paddack
Lawrence Livermore National Laboratory
7000 East Avenue, L-644
Livermore, CA 94551
(925) 422 0864
fax (925) 423 7914
Saceves@llnl.gov

Paper submitted to the 9th World Congress on Intelligent Transportation Systems
October 14-18, 2002
Chicago, Illinois

SUMMARY

We are working on developing an Integrated Systems Analysis Environment (ISAE) for application to analysis and optimization of Intelligent Transportation Systems (ITS). ISAE is based on the concept of Co-simulation, which allows the modeling of complex systems with extreme flexibility. Co-simulation allows the development of virtual ITS systems that can be analyzed and optimized as an overall integrated system. The virtual ITS system is defined by selecting different components from a component library. System component models can be written in multiple programming languages running on different computer platforms. At the same time, ISAE provides full protection for proprietary models. Co-simulation is a cost-effective alternative to competing methodologies, such as developing a translator or selecting a single programming language for all system components. Co-simulation has been recently demonstrated using an example of an automotive system. The demonstration was successfully performed. The paper describes plans on how to implement ISAE and Co-simulation to ITS, and the great advantages that this implementation would represent.

INTRODUCTION

The design, development and deployment of Intelligent Transportation Systems (ITS) requires the successful integration of many disparate building blocks, including subsystems for sensing, control, communication, computation and display. These components may be in vehicles, connected with travelers themselves, on roadways, or in fixed-location centers. Figure 1 shows high-level ITS functions defined by the National ITS Architecture. Each of these functions, when present in a system, consists in turn of numerous components interconnected with each other at finer levels of detail.

To facilitate seamless connection of components into an effectively functioning overall system, the US Department of Transportation (USDoT) has established detailed architectural and communication specifications for the many functions imbedded in Figure 1. In many cases, standards are also being established for specific interconnected subsystem types. As these standards become accepted and incorporated into commercial products, the problem of integrating components should become easier. However, because of the complexity of real subsystems, the enormous number of ways they may interact with each other and with their environment, and the inevitable differences in technology among various suppliers, the integration process will never become completely routine. Furthermore, because most communities have a substantial existing investment in legacy systems, there will continue to be a difficult integration problem in most affordable upgrades of ITS systems for some time to come (1).

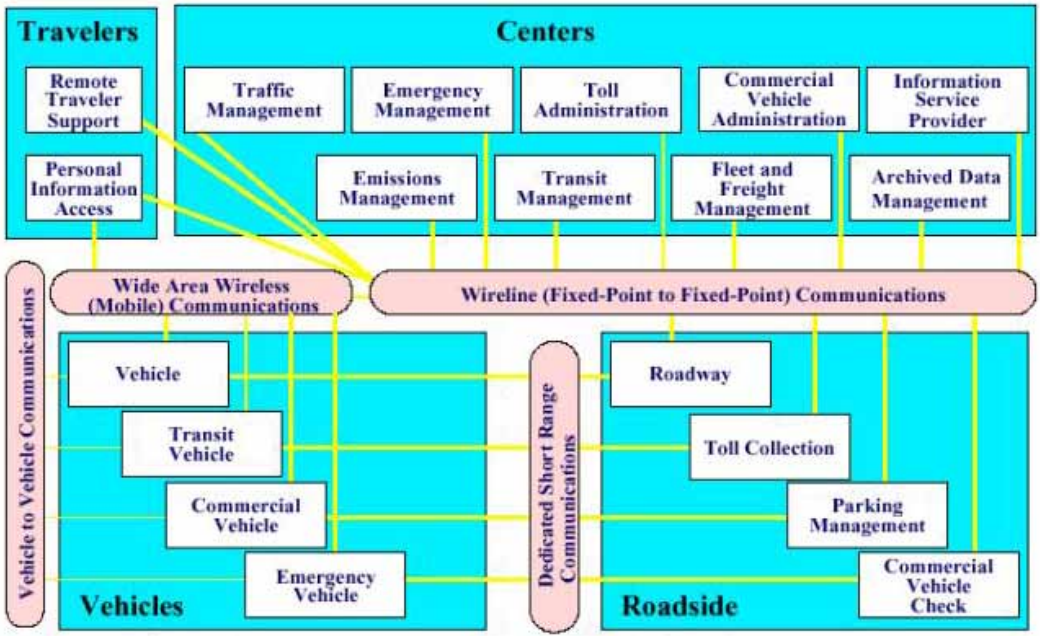


Figure 1. High-level ITS functions, as defined by the National ITS Architecture.

At present, system engineers integrate ITS systems based on a combination of hard-earned knowledge and experience, on data from manufacturers, and on testing interconnected system components. This laborious process is costly in time, money, and potential performance degradation. In an era when it is critical to demonstrate cost-effective solutions to increasingly onerous traffic problems, public confidence can be easily undermined by cost overruns, implementation delays, and performance losses. For all these reasons, it would be extremely desirable to have an efficient simulation environment for testing ITS deployments prior to hardware purchase decisions and expensive prototype testing. The Integrated Systems Analysis Environment (ISAE) offers the promise of such a tool.

Note: This tool addresses the primary problem of assembling an integrated ITS system that functions effectively in an engineering sense, not in a traffic management sense. Traffic simulations, at some appropriate level of detail and realism, would be a part of the system simulation, but primarily as drivers of the sensing, control, communication, and control components. This tool does not directly address the question of the effectiveness of the system from a transportation point of view, although it would be a useful adjunct for such analysis. Its most critical role is the necessary one of testing whether the system can successfully and reliably perform all the functions that the traffic engineers require.

A SYSTEMS INTEGRATION ENVIRONMENT

We are planning to develop the Integrated Systems Analysis Environment (ISAE) into a powerful platform for large-scale systems analysis and development. Using this environment, third parties and end users can easily configure, simulate, and build systems

from modular building blocks. The flexibility and power of ISAE derives from the fact that it is independent of the language and computer system used by each building block, and respects the proprietary content of individual components.

ISAE is based on the concept of Co-simulation. Co-simulation is a relatively new approach. In Co-simulation, only input and output data are transferred between models under the coordination of a Commercial Off The Shelf (COTS) Co-simulation software package.

Previous approaches for simulation of complex systems have included the use of a single programming language or the use of translators. Models developed using a standard language will, in most cases, interact with each other. However, existing models are typically developed using different modeling languages favored by the individual code developer or the application. Rewriting all models to a standard language is a monumental undertaking for the following reasons:

- Difficulty in obtaining consensus on the selection of a standard modeling language due to participants' own familiarity and preference.
- Rewriting takes a lot of time; costs are prohibitive.
- Re-validation of new models is necessary.
- Different modeling languages are more suitable for particular situations.

Another common solution is to build a translator that can understand all modeling languages in use and translate all sub-system models into standard executable codes. These code modules can then be linked and executed by the simulation engine. In this case, existing models are not modified. However, the translation process invalidates the validation of the original models and the code modules or the translation process must be validated again. The disadvantages of building a general translator are:

1. Building a translator and the simulation engine is a very complex undertaking and involves significant effort and cost.
2. It is very time consuming.
3. It requires validation of the translation process, the resulting code modules and the simulation engine.

Co-simulation does not require any modification or translation of the original models. Better yet, it does not even require access to the source code of the model. This is a very important issue in protecting proprietary information and intellectual property rights among project partners in the competitive market. The Co-simulation methodology is illustrated in Figure 2. The advantages of this approach are many:

- It is extremely flexible: complex system models can be executed in a multi-platform, multi-language, over an Internet environment.
- Relatively inexpensive because no rewriting or translation of existing models is needed. This makes Co-simulation the most cost-effective option. This is an

overwhelming advantage that preserves existing model integrity and substantial savings in time and expenses.

- No interference to existing models in terms of ownership, control, and proprietary protection.
- No modification or re-validation of existing models is needed.
- The integration is at the modeling language (tools) level, which eliminates the need to deal with the actual models.
- Co-simulation is applicable in all levels of organizations: project, group, department, corporation, business partners, and industry.

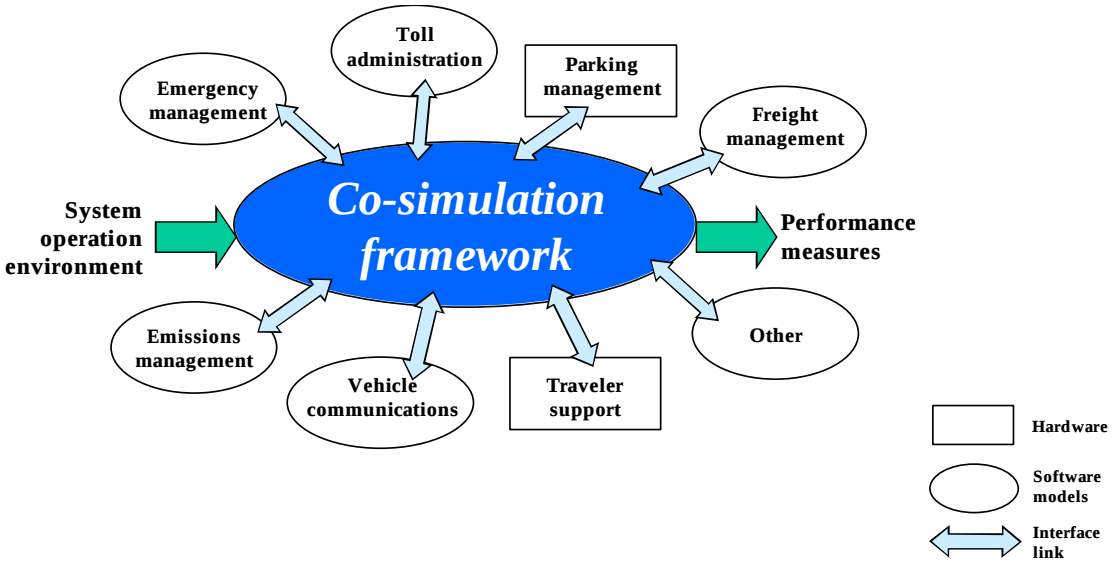


Figure 2. Example of an overall ITS systems architecture employing Co-simulation.

The basic idea behind Co-simulation is the use of software components, which are pre-built, pre-compiled, pieces of software. Instead of coding a control application from scratch, it is assembled from these components. To make assembly of these systems possible, the components implement pre-defined, standardized sets of services, and are referred to as the Application Programming Interfaces (API). Components may use the services provided by other components in order to provide their services. When using the services of another component, the actual implementation details of that service are not important; it is the API that accesses that service which is critical.

AN ITS EXAMPLE

Creating a Traffic Management System simulator requires software components to be available. These software components represent each physical component, e.g., for traffic, traffic sensors, controllers, communications, traffic management displays, traffic managers, etc. A Co-simulation-defined API would need to be defined to represent each

type of connection between these components. From the outside, each of these APIs appears to be a "black-box." On the inside, these simulation components could be complex or act as a thin "front-end" for an existing simulation model.

Each component could be simulated by different, pre-existing simulation packages, providing that all the packages could be made to work with either the same sample time-step or a multiple of the smallest time step required. None would actually have to perform their calculations in real-time.

In addition, each pre-existing simulation would require a way for the Co-simulation software to get data in and out of the model, using the APIs defined. It is likely that much of the work of the National ITS Systems Architecture will be applicable to defining components, interconnections, and APIs.

SUMMARY OF THE TECHNOLOGY

The ISAE technology vision provides for easily customized plug-and-play components for complex system simulation to reduce cost and provide higher fidelity while leveraging pervasive, off-the-shelf, high-volume, software technology. Technology requirements seen as critical to the success of ISAE include:

1. Facilitate component-based plug and play analogous to PC industry
2. Maximize design and component reuse to reduce cost
3. Maximize component customization to allow commodity valuations of reusable components gained from leveraging cross-industry purchasing power
4. Enable the trend towards smart "introspective" components that contain the knowledge previously found on bookshelves
5. Allow component use at design time in an Integrated Development Environment (IDE) as well as runtime
6. Provide flexibility to handle many applications with similar solutions
7. Allow extensibility to accommodate new technology
8. Ensure modularity to encapsulate and isolate changes

SUMMARY OF BENEFITS FOR ITS

An ISAE would do for ITS simulation what the National Systems Architecture does for ITS systems themselves: support seamless integration of system models. It would help assemble existing models into functioning simulations, and it would also be the basis for standards and templates upon which vendors could build models for easier interconnection in the future. The basic benefit, of course, would be to streamline the process of system integration, and thus facilitate the deployment of ITS systems nationwide.

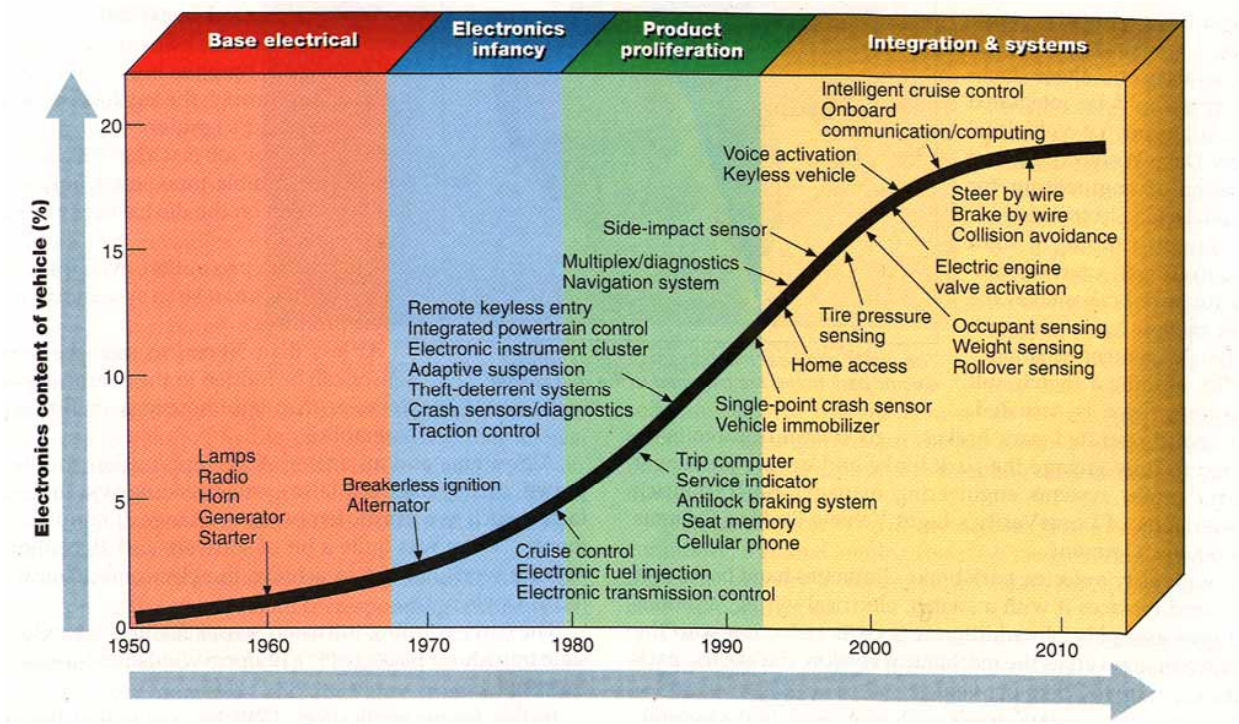
A CO-SIMULATION PROOF-OF-CONCEPT

Co-simulation is a relatively new technology. This makes it important to conduct a Proof-of-Concept demonstration to verify that a working system can be developed. It is also necessary to test the commercial off the shelf (COTS) software available in the market place for applicability to ITS. We have recently conducted this demonstration using as an example the integrated simulation of vehicle components (2).

In many respects, vehicular systems are similar to an ITS system, where different components interact with each other in complex, non-linear fashion (Figure 2). The methodologies applied for Co-simulation of vehicle components are therefore directly applicable to ITS systems.

Vehicle designs are becoming more complex and sophisticated. Industry sources estimate that up to forty percent of the production cost of a present-day luxury car is allocated to electronics and electronic control systems (3). This allocation is forecasted to increase rapidly. For example, parts suppliers will provide an ever-increasing array and variety of electronic and electromechanical components, and any selected component must be rapidly incorporated into the overall design in the most efficient manner possible. The issue of component controls interfacing, communication protocols, flexible data bus, and multiplexing architecture become critically important. Close coupling of many distributed electronic control systems will also be required, and different equipment suppliers will develop many candidate solutions. Figure 3 shows the trend of electronics content per vehicle (3).

The ability to efficiently model and simulate interactions between vehicle components is essential for the accelerated development of more fuel-efficient and safer vehicles with reduced emissions. Currently there does not exist a universal modeling and simulation capability that addresses, in an integrated fashion, all vehicle electronic and mechanical subsystems. There is also no capability that captures the interactive effects between vehicle electronics and functional subsystems. Rather, individual subsystem simulations are independently performed using various commercial software packages, but the increasingly important integrated controls issue is not being adequately addressed.



Electronics content per vehicle.

Automotive Engineering International/September 1998

Figure 3. Growth of electronics content in new vehicles as a function of model year.

A project team was assembled with personnel from LLNL, WindRiver, Avant! and Sun Microsystems, with input from General Motors Research (GMR). Several automotive systems were considered for the proof of concept demonstration. Finally, it was decided to use a complex model of an automobile transmission written in SABER. The transmission controller was removed from the SABER model and re-coded in MATRIXx. MATRIXx was run on a PC running Windows NT, while SABER was run on a four-processor SUN/Ultra 80 running SOLARIS (Figure 4). The COTS product chosen for the prototype and demonstration is 'pLUG&SIM'; a software product released by Integrated Systems, Inc., which later became WindRiver Corporation. The computers were connected through a network hub using TCP/IP protocol. The team met for the first time two days before the scheduled demonstration. We were able to overcome numerous expected and unexpected hardware and software problems and succeeded in a flawless demonstration of a multi-host, multi-tool, Co-simulation seen for the first time in public in Detroit. The proof of concept demonstration was successfully conducted. The demonstration provided a crucial proof of concept for the technical approach by demonstrating a multi-host, multi-tool Co-simulation.

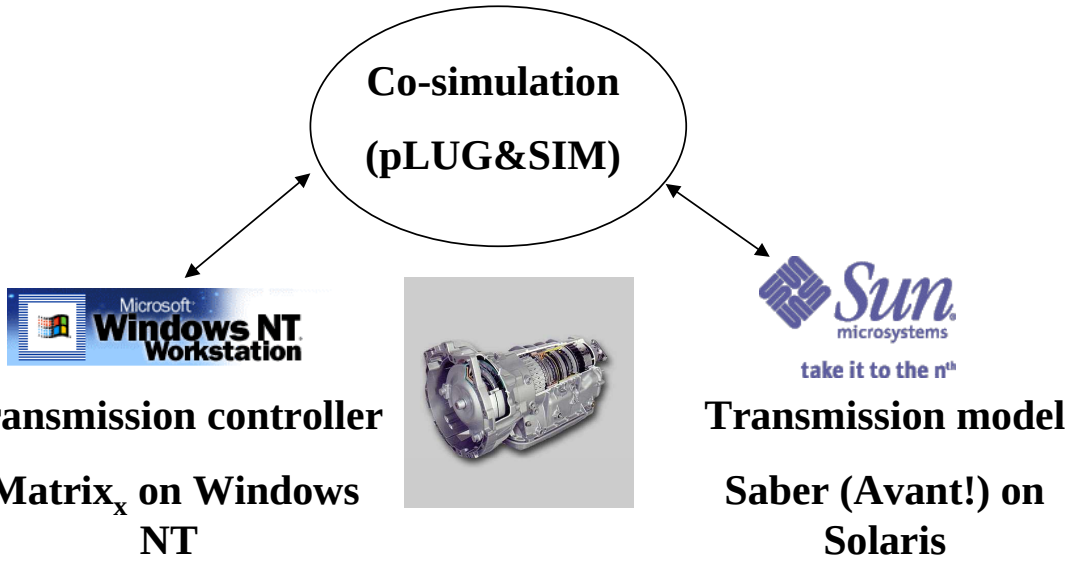


Figure 4. Software and hardware configurations used for the proof of concept demonstration of the Co-simulation concept.

PLANS FOR FUTURE WORK

We plan to extend the basic Co-simulation concept to cover the domain of ITS systems analysis. The framework developed for automotive Co-simulation is easily extended to include time-based simulations. The future effort falls into two broad categories.

1. Extension of the Co-simulation architecture for systems analysis.
2. ITS-specific APIs and a reference implementation of those APIs.

Specific steps include the following:

1. Select a specific ITS deployment type. Traffic management appears to be a promising target application.
2. Establish a collaboration with an implementation project. The I-580 Smart Corridor appears to be an excellent target of opportunity, with LLNL already involved as a partner in the areas of system testing and validation.
3. Review the National System Architecture (NSA), and borrow as much as possible from it. Use it as a basic guide to the system model, with respect to component inputs and outputs and interconnections. Base the structure and implementation of the new tool as consistent as possible with existing NSA software applications.
4. Review key relevant ITS components and existing models, with respect to availability, language, platforms, type, etc.
5. Modify and/or build models as needed.
6. Assemble an ISAE for the application.
7. Apply the ISAE to the test application.
8. Provide the ISAE template to other users for evaluation.

ACKNOWLEDGEMENTS

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

REFERENCES

1. Bruce Abernethy, “Crossed Lines,” Traffic Technology International, April/May 2001, pp. 40-47.
2. Walter Ng, Erma Paddack, Salvador M. Aceves, “Optimum Vehicle Component Integration with InVeST (Integrated Vehicle Simulation Testbed),” Proceedings of the ITS America Conference, Long Beach, CA, April 2002.
3. Kami Buchholz, “Electronics Innovations,” Automotive Engineering International, Vol. 106, No. 9, September 1998, pp. 35-38.